情報システムとWeb技術'15 ＃1: 情報システムの定式化; HTML+CSS

久野　靖 *

2015.11.17

0　はじめに

本科目「情報システムとWeb技術」は、今日の世の中の基盤(インフラストラクチャ)となっている情報システムについて、それがどのようなものであり、どのようにして動いているのかについて、使用者側からの視点だけでなく、発注者・設計者・構築者の視点まで含めて、概要を理解することを目的としています。といっても、このような遠大なテーマをたった5回で詳しくカバーすることなど到底無理ですから、本科目では次のような方針を採用します。

まず、情報システムの(構想・設計・構築の)全体的な考え方については、各回とも重くなりすぎない範囲で側面を絞ってトピック的に取り上げることにします。そしてその残りについては、個別の題材について例題を用いて技術的な側面を中心に見て行きます。具体的な題材としては、現在の情報システムにおいて重要な位置を占めるWebシステムを用い、その実装の個別の側面に絞って取り上げて行くことにします。ここが「…とWeb技術」に相当するわけです。

個別の例題については、実際に例題コードを動かしてみながら、それを参考に自分なりに手直ししたり新たなコードを追加したりして、課題をこなして頂くようにします。今年度はコードはSQL(データベース操作言語)とPHP程度を中心としていくように考えました。

各回において取り上げるテーマ・内容ならびに目標としては、とりあえず次のように予定します(今回、科目内容をだいぶ改訂しているため、実施して見ないと分からないところもありますから、これはあくまでも予定です)。

＃1 情報システムとその定式化

- 情報システムの定義とその役割/ネットワークの位置づけ
- 情報システム定式化の手法と構造化分析/データフロー図
- 業務分析のモデル、IDEF1Xによる記述

目標: 情報システムの定式化・モデル化について基本事項を理解する。DFDやIDEF1Xなどのモデル図が描けるようになる。

＃2 データベースとDBMS

- データベースとは/情報システム中でのデータベースの位置付け
- 関係データベースとその諸概念
- データベース管理システム(DBMS)とその機能
- データ操作言語SQLとSQLによるさまざまなデータの操作

目標: 情報システムの基盤であるデータベースを理解し、関係データベースの操作が行えるようになる。自分の必要に応じ使えるようになる。

*経営システム科学専攻

＃3 WWW と Web アプリケーション

- WWW の基本構成、ブラウザと Web サーバの機能、HTML/CSS
- Web アプリケーションの構造と動作
- フォーム入出力、PHP による Web アプリケーション
- セッションとセッション管理、安全性の管理

目標: Web 技術 (HTML/CSS、Web サーバの役割、Web アプリケーションの基本) について理解する。簡単な Web アプリが書けるようになる。

＃4 Web アプリケーションとユーザインタフェース

- 人間が持つ認知特性とユーザインタフェースの関係
- Web アプリケーションにおけるユーザインタフェース設計の役割
- ユーザインタフェースの設計手法/評価手法
- ユーザエクスペリエンスの概念

目標: ユーザインタフェースの基本原理やデザイン手法について理解しする。ユーザインタフェースを考慮した Web サイトが作れるようになる。

＃5 データベース連携 Web アプリケーション

- セッション管理の具体的手法
- データベースと Web アプリケーションの連携方法
- 製作実習

目標: データベースと Web アプリケーションの連携について理解する。データベースを扱う Web アプリケーションの仕組みについて理解する。

本科目の評価については、各回の出席 (議論・実習への参加) プラス最終レポートによるものとします。最終レポートは、各回の資料に含まれているさまざまな課題から好きなものを 1 つ以上選んでレポートを書いて頂くものです。

では、これから 5 回にわたり、よろしくお願いします。

1 情報システムとその定式化

1.1 情報システムの定義とその意義

システム (system、体系) とは、全体としてまとまった機能を実現するような、構成要素の有機的な集まりを意味します。そして情報システムとは、情報を取り扱うことを目的としたシステムを言います。

たとえば人間や昆虫などの生物の場合、情報を扱うことがその働きの大きな部分を占めることは確かだがが、情報を取り扱うことを目的としているかどうかについては疑問があるので、普通は情報システムであるとは言わないでしょう。諜報機関のような組織について考えれば、確かに情報を扱うことを目的としたシステムであるので、上の定義にあてはまります。

しかし今日では、最も広く見られる情報システムは、コンピュータを中心としたシステムということになるでしょう。というのは、コンピュータは情報を取り扱うことを目的とした装置であり、しかもその出現以前には例が無かったほどに (人間の手作業による処理では不可能なほどに)、大量の情報を正確・高速に取り扱うことを可能にしているからです。

人間を他の動物を制して地球の支配者たらしめたものは、人間が持つ情報処理能力であることは間違いありません。そして、人間が持つ情報処理能力を (ごく単純な部分に限られているとはいえ) 代替し、かつ増強することを可能にした情報システムというものが、人間の存在にとって重要なもので

あることもまた確かです。そもそもそんな理屈を言わなくても、私たちは日常的に多くの情報システムに囲まれ、その恩恵を受けながら生活しています。

質問 1 今日私たちの身のまわりにある情報システムとして、具体的にどのようなものがあるか、それは情報を「どのように」扱っているかをいくつか思い付く範囲で列挙してみなさい。

1.2 情報システムの定式化

では次に、ある情報システムがあったとして、それはどのように定式化できるでしょうか。もっとひらたく言えば、それがどのような構造を持ち、その個々の要素がどのような働きを受け持っているかを、きちんと表現するにはどのようにしたらよいでしょうか。

もちろんそのために、先人がさまざまな手法を考案したり定式化したりしてきています。その中にはZのような論理式に基づく (形式的手法に近い) ものもありますが、ぱっと見分かりやすいのは図を用いた方法でしょう。

図による記法もまた色々な流儀がありますが、ここではデマルコによる階層化されたデータフロー図 (DataFlow Diagram、DFD) を紹介します。デマルコが提唱した手法は「構造化分析 (Stractured Analysis)」と呼ばれ、システムを階層的に分解して「どのようなシステムであるか」を同定する (分析する) ことを目的とした手法です。DFD はそのための記法なのですが、今では図法である DFD の方が有名になっています。

質問 2 あなたはシステムを記述し分析するのにどのような手法を使っていますか。またはどのような手法を知っていますか。その手法の利点や弱点は何ですか。

1.3 構造化分析入門

構造化分析の DFD では、○でプロセス (何らかの処理) を表し、矢線でデータの流れを表し、線分でデータの蓄積場所 (ファイルや DB) を表します。構造化分析では基本的にはこの 3 つと、あとシステムの外部にあるエンティティを表す□だけで、情報システムの構造を記述します。図 2 に簡単な DFD の例を示します。

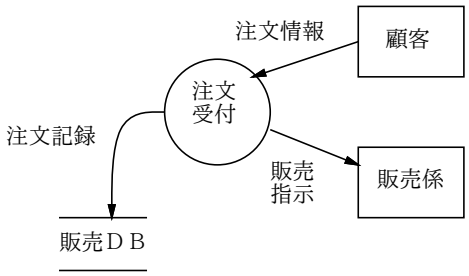

図 1: 簡単な DFD の例

これは何らかの商品の在庫を管理し、顧客からの注文に応えて出荷を行うようなシステムを表しています。ここで「商品入荷」プロセスから「受注」プロセスに「入荷通知」データの流れがある理由が分かるでしょうか。

これはつまり、顧客から注文を受けた時に在庫がなければ、そのことを在庫記録データに記録しておき、商品入荷時に顧客からの受注残のある商品が入ったとわかったらそのことを受注プロセスに知らせる必要があることを意味しています。

しかしそうすると、「受注」プロセスは入荷通知があったときにその商品を予約注文している顧客のデータをどこからか持って来なければなりませんが、そのアテがありませんね? ということは、その情報を格納するデータベースを図に追加する必要があります。

このように、構造化分析では (他の分析手法もそうですが) システムを図で表現して検討することで、見落としや不整合や矛盾を発見して、正しいシステムの構造を作り出せるわけです。

DFD は構造化分析と呼ばれるシステム分析手法のために考案されたものですが、直観的に分かりやすく広く使われています。ここでは構造化分析とかは省略して、DFD の記法と使い方だけ説明します。

DFD には次の 4 種類の要素が含まれています。

- 処理 (プロセス) — ○で表され、コンピュータによるデータの処理を意味します。通常、1 つの情報システムの中にはさまざまなデータ処理が含まれますが、それぞれを別の○で表します。
- 外部主体 (エンティティ) — □で表され、システムの外部にあって、データをシステムとやりとりするようなものを意味します。なぜこのようなものがあるかというと、情報システムは必ず外部とデータをやりとりするからです (さもないと役に立たない)。
- 記憶場所 — 水平線 (1 本か 2 本) で表され、データを蓄積・保管する場所を意味します。一般にはデータベース (db) を意味しますが、いわゆる「台帳」をイメージして頂いて構いません。
- 矢線 — データの流れを表し、上の 3 つの要素を互いに結びます。それぞれの矢線には「どんなデータが流れる」かを必ず記します。

たとえば、図 2 に「湧き水」販売会社のシステムを掲載します。この会社は自社が所有する霊験あらかたな湧き水を 1 杯 10 円で販売しており、その記録のためのシステムです。商品は無尽蔵に湧いて来るので、顧客から注文があったらそのことを記録し、販売係に指示を出せばそれでいいわけです。

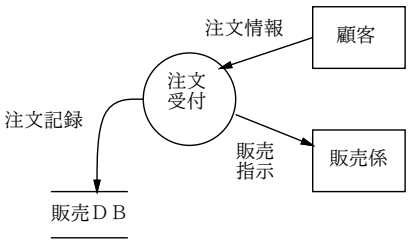

図 2: 「湧き水」販売システムの dfd

ごく単純そうに見えますが、次のことを考えてみてください。

- 「注文受付」処理はどのような動作を行うか。
- 「販売情報」にはどんなデータが含まれているか。
- 「販売指示」にはどんなデータが含まれているか。
- 「注文記録」にはどんなデータが含まれているか。

こういうことをきちんと決めないと、情報システムは構築できないわけです。

そして、ただ単に「頭で考えている」だけでは見落としや勘違いが生じやすくなりますが、dfd のような図法を用いることで「全体の構造を見る」「それぞれの箇所に注目してチェックする」などの作業が行いやすくなります。また、図法が決まっていることで、「どのような図を描いたらいいか」という悩みもある程度緩和されるわけです。

さて、「湧き水」販売システムのデータを仮に定義するとしたらどうなるか、というのを考えてみたものが図 3 です。お客さんには番号 (食券の番号のようなもの) を付与するとして、何番のお客さんは何杯、というのが注文情報となり、それを販売係にそのまま渡します。また、記帳のため販売 db に記録するものは、これらの情報に日付と時刻を追加したものでいいのではないでしょうか (ここでは注文記録と販売 db は同じ内容になりますが、販売 db に書かれるのは注文記録のデータしか無いわけですから同一で当然です)。

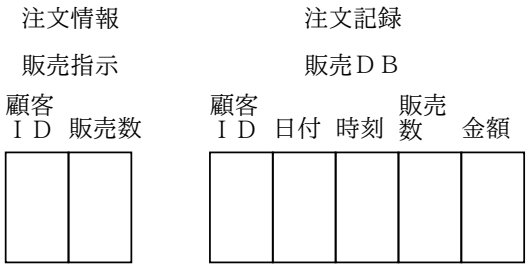

図 3: 「湧き水」販売システムのデータ定義

1.4 簡単な演習: 在庫を管理する場合

湧き水の販売では商品は無尽蔵にあったのですが、普通はそうではなく、お店に商品 (ここでは食品とします) がストックしてあって、注文があれば売り、在庫が少なくなったら問屋から仕入れます。そのようなシステムの dfd の例を図 8 に示します。

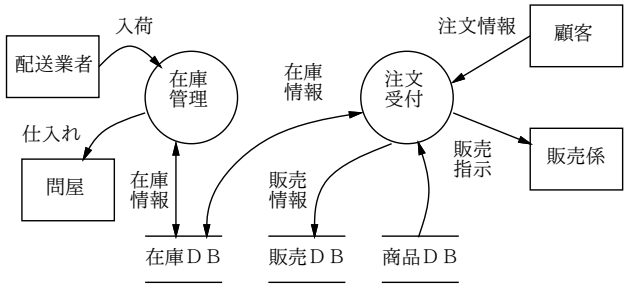

図 4: 在庫のある販売システムの dfd

演習 上記の DFD について、次のような場合それぞれについて、DB やデータの内容がどうなるか検討してみなさい。db を追加する必要があるかも知れませんが、その場合はどのような DB かも併せて検討しなさい。

a. 商品は 1 種類であり、問屋に注文すると瞬時に配送される。
b. 商品は数種類であり、問屋に注文すると瞬時に配送される。
c. 商品は数種類であり、問屋に注文するとどれでも 1 時間後に配送される。
d. 商品は数種類であり、問屋に注文するとどれでも 3 日後に配送される。
e. 商品は数種類であり、問屋に注文すると注文ごとに配送までの日数が変動する。

1.5 やや複雑な演習: 自動販売機の売上・補充管理

ではいよいよ、本題の演習課題の説明に進みます。駅などで、自動販売機のドアをあけたところを見たことがあると思います (図 5)。販売機の中は「カラム」と呼ばれる缶を入れる箇所が並んでいて、それぞれのカラムに別の飲料が入っています。

すべての飲料を沢山持っていってこまめに巡回し、毎回目一杯補充すれば情報システムは無くても済みますが、膨大なコストが掛かります。今回想定する販売機は通信機能がついており、売れるたびに「何番のコラムが残り何個になった」というデータを送信できるものとします。そのような販売機を 1000 台くらい運用している企業から、「情報システムを導入して、補充を効率化し、売り上げ増につながる施策も打ち出せるようにしたい」というざっくばらんな要望を受けたものとします。

皆様のチームは次の 3 種類の役割の人が入っていることにします。

- システム設計者 (se) — システム全体の構造や機能を整理し、なるべくシンプルで顧客満足の高いシステムを設計したい。

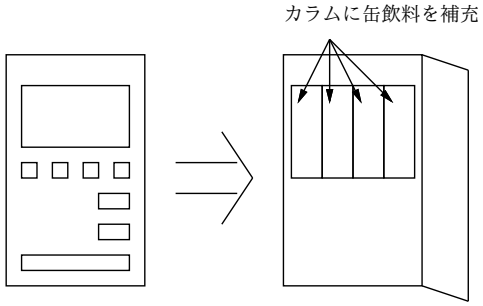

図 5: 自動販売機の内側

- 顧客企業代表 — 企業利益を最大化したい。それにはなるべく低いコストで事業を運用することがまず考えられるが、運用コストが上昇してもそれ以上に利益が上がる方法があるなら、そちらを選択したい。もちろん、システムの開発や運用にかかるコストも少なくしたい。
- 営業担当代表 — まず各販売機の状況が分かって適切に補充ができることが最低線であるが、現場での気付きの活用や商品の適切な入れ替え計画への活用など、営業立案に役立てられるとなおよいと考えている。

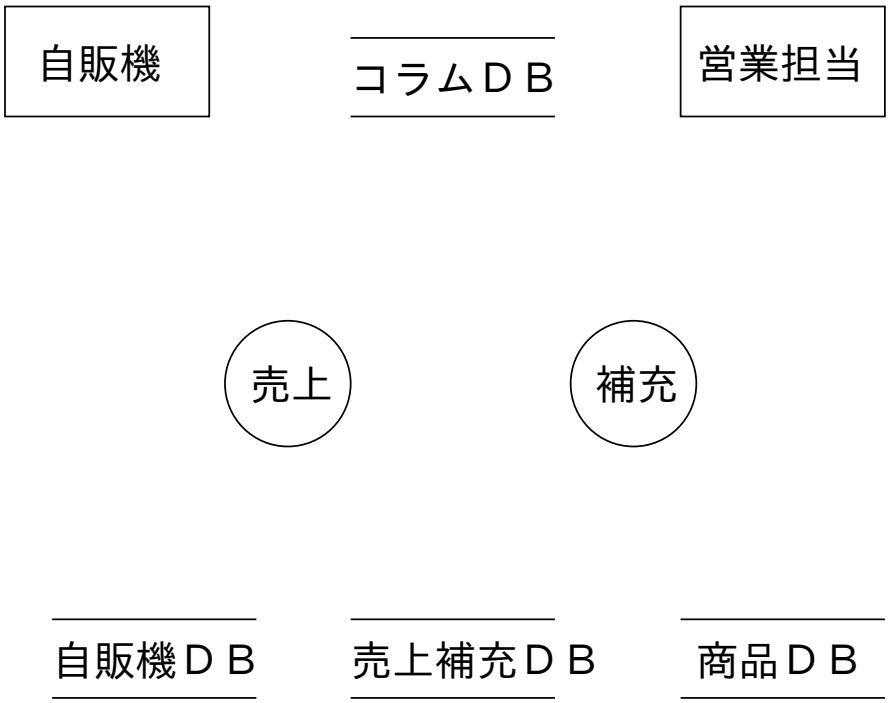

図 6: 最低限の要素を示した dfd

図 6 に、最低限の要素 (データ、処理、外部主体) を描いた dfd を示します。データの流れは描いてありませんし、データの中身ももちろん描いてありません。[1] これを印刷した紙と「収集した情報のカード群」を印刷したものを、別途配布します。

演習 演習は次の手順で進めてください。

- グループで se、顧客代表、営業担当代表 (それぞれ 1〜2 名) を決め、ポストイットで役割を書いた名札をつくって胸に貼る。

[1] 1 つだけヒントがあります。「コラム db」って何だろうと思ったかも知れませんが、自販機 db には各の自販機の全体としての情報だけがあり、その自販機の中の各コラムに関する情報は「コラム db」に入れます。これは関係データベースの制約から来るものです。

- カードを 1 枚ずつ皆で確認する。そのカードの発言は誰の発言ということにするか決めて各自書きこんでください。
- 皆で db の内容、データの流れのつながり、データの流れの内容を検討し、書き込んで行く (書き込む用紙は余分にあるので書き損じたら取りに来てください)。
- ひととおりできたらその設計で大丈夫そうか全員で確認する。
- まだ時間があるようだったら、顧客代表や営業担当代表は「もっとこういうことができたらいいのだけれど」と我儘 (?) を言い、さらに検討して改良版の案を考える。

演習 次のものからどれかを選び、その主要な構成要素 (プロセス、データベース) を記述する DFD を描いてみなさい。

a. (たとえば TWINS のような) 大学の成績管理システム
b. (たとえば首都高の) 道路情報システム
c. (たとえば楽天トラベルのような) ホテル予約システム
d. (たとえば東京メトロの) 券売機の制御システム

1.6 階層化

前節では 1 枚の DFD でシステム全体を記述できるかのような説明をしましたが、もちろん複雑なシステムではそんなことは無理です。大きな紙で無理矢理描いたとしても、ごちゃごちゃで何が何だか分からなくなるのがせいぜいです。

構造化分析ではこの問題に対して、DFD を階層化することで対処しています。まず、システム全体を 1 つのプロセスとして描いたダイアグラムをコンテキストダイアグラム (図 7) と呼びます。これは、システムが外部とどのようなやりとりをするのか、システムに関与する外部者は何であり、システムからは何が出入りするのかを網羅する役割があります。

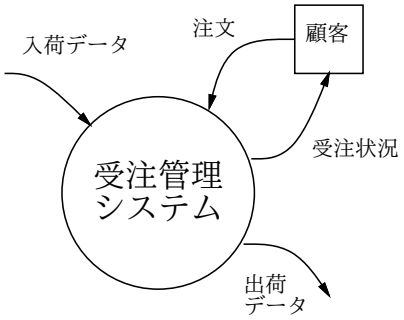

図 7: コンテキストダイアグラム

これを分解した中身が先の図 2 で、これがトップレベルの (レベル 0 の) ダイアグラムとなります。そして、そこに出て来る○ (プロセス) は通常、さらに細かい処理に分解されます。たとえば、1 番の「商品入荷」のプロセスは、さらに 3 つの処理に分解されるかも知れません (図 8)。

こうすることで、上のレベルでは全体的な構成をチェックでき、また下のレベルでは細かい処理の割り振りをチェックできるわけです。そして、必要ならこの各プロセスもさらに細かく分けて行き、具体的な手順として実現できる程度まで分けたら、あとは設計に進んでコードを書けるようにして行きます。

一般にコンピュータサイエンスでは、階層化によって抽象化の高いレベルから低いレベルまでを統一的に扱うという定石がありますが、ここでもそれが適応されているわけです。

なお、このような DFD の階層分割に際しても、守るべき要件があります。

- 下のレベルの処理が全体として、上のレベルの 1 つの○の処理を過不足なく実現していること。

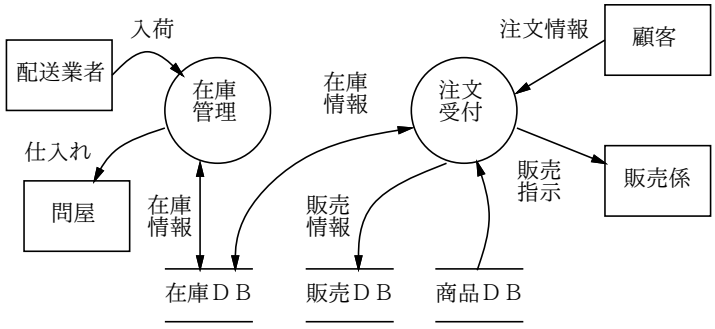

図 8: レベル 1 ダイアグラム

- 上のレベルの○に現れるすべての矢線は、下のレベルにも対応してどこかに現れること。下のレベルに、上のレベルには対応するものが無いような矢線が現れないこと。

各レベルにおいてこのようなチェックを経ていくことで、整合性があり間違いのできるだけ少ない分析をおこなう、というのが構造化分析の要点なわけです。

構造化分析については、デマルコの次の本が定番です (原著は非常に古い本ですが、日本語版は今でもよく売れていて増刷されています)。

> トム・デマルコ著, 高梨・黒田監訳, 構造化分析とシステム仕様, 日経 BP, 1994.
> (原著: Tom Demarco, Structured Analysis and System Specificaion, Prentice-Hall, 1979.)

この本には分析とは何か、構造化分析とは何かからはじまってとても丁寧に分析の進め方が説明されています。たとえば、それぞれの○の中を説明する用語は辞書を作って管理して未定義がないようにするとか、○の中の処理を手続きで実現する (構造化設計) とかまであるのですが、今だとどこかから先はオブジェクト指向でしょうし、そのあたりはちょっと古くなってるかな、と思います。しかし分析自体については今でもまったく古さを感じさせない名著ですので、読んでおくことを薦めます。

演習 前問で作成した DFD について、コンテキストダイアグラムを描き、また階層分解してみなさい (どのレベルまでやるかは適当に決めてよいです)。

2 HTML+CSS

2.1 WWW とマークアップ言語

マークアップの理屈は分かったけれど、やはり見たまま (WYSIWYG) 方式の方が分かりやすいし使いやすい、と思われたかも知れません。しかし実は、見たまま方式が「使えない」場合というのも存在します。しかも、皆様がいつも目にしているもの — Web ページがまさに「そういう場合」なのです。なぜだか分かりますか?

ワープロであれば、出力する紙サイズが決まっていて、その紙に合わせて配置や文字のサイズを決めて行くことで「見たまま」を自由に調整できます。しかし、Web ページはどうでしょうか? Web ページには「紙サイズ」は存在しませんから、「1 行何文字詰めで文書を作る」ことは不可能です。マシンによって使えるフォントも文字サイズも変ってきますから、「この表題は MS 明朝の 24 ポイント」と指定しても、そのフォントがないかも知れません。

ではどうすればいいのでしょうか? できることは「ここは表題」「ここは段落」「ここは箇条書き」などと「構造を」指定しておいて、「ブラウザが画面に表示する時に」窓の幅や使えるフォントに合わせて整形してもらうことです。つまり意味づけ方式でマークアップするしかないわけです。

「でも私はWYSIWYGツールでWebページを作っているが」という人もいるかも知れません。しかし、実はそれは「WYSIWYGみたい」なだけで、本当のWYSIWYGではないのです。というのは、あなたが使っているマシンと表示能力や画面サイズの違うマシンに行ったら、どのみち「その通りに」表示することは不可能なのですから。

なお、WYSIWYGツールが無意味だというつもりはありません。とりあえず「自分のマシンならこんな仕上り」という様子を見ながら編集できるのはそれなりに便利だと思います。しかし、Webアプリケーションなどでプログラムから HTML を生成する場合には、どのみち直接 HTML を扱う必要がありますから、HTML+CSS についてどのようなものかを知っておくことはやはり必要でしょう。

2.2 HTML の概要

HTML(HyperText Markup Language) については前回も既にやったので、ここではごく簡単なまとめから始めます。HTML ではマークアップの部分を**タグ** (tag) と呼び、マークアップで囲まれた文書の範囲のことを**要素** (element) と呼びます。要素の構成は次の 2 種類です。

<名前 オプション…> …内容… </名前> ←タグが対になる要素
<名前 オプション…> ←単独タグだけの要素

HTML の用語では「オプション」の部分を**属性** (attribute) と呼びます。次にごく簡単な HTML を再掲しておきます。HTML が指定するのは文書の「構造」だということを確認しましょう。

```
<!DOCTYPE html>
<html>
<head>
<title>○○'s page</title>
</head>
<body>
<h1>○○です。</h1>
<p>…挨拶ないし自己紹介を書く…</p>
</body>
</html>
```

ここにあるものも含め、主要な HTML 要素には次のものがあります。

- <!DOCTYPE html> — 以下が HTML 文書であることをあらわす。
- <html>…</html> — HTML 文書全体をあらわす。
- <head>…</head> — ヘッダ (この文書に関する情報を記述する部分) をあらわす。
- <title>…</title> — 文書のタイトルをあらわす。
- <body>…</body> — 文書本体 (ブラウザの窓の内側に表示される内容全体) をあらわす。
- <h1>…</h1> — レベル 1 の見出し (大見出し) をあらわす。さらに小さいレベルの見出しとして<h2>…</h2>〜<h6>…</h6>まで使うことができる。
- … — 通常の段落をあらわす。
- <pre>…</pre> — 要素内の空白や改行をそのままにする。
- … — 要素の範囲を強調表示する。☆
- … — リンクをあらわす。☆

HTML では「<」、「>」、「&」は特別な意味があり、そのままでは使えません。代わりに「`<`」、「`>`」、「`&`」と書いてください。

なお、本体 (body の中) に使う要素には、**ブロック要素とインライン要素**の 2 種類があります。ブロック要素は段落相当のものであり、その前後では改行されますし、段落 (p 要素) 内・インライン要素内に入れられません。一方、インライン要素は段落内の文字範囲に相当するので、前後で改行されず、段落や他のインライン要素内に入りますが、ブロック要素を囲むことはできません。ここまでに出たインライン要素は☆のついたもの (a 要素と em 要素) だけです。

2.3 構造と表現の分離、スタイルシート

ここまで「HTML は文書の構造を規定する」と説明してきました。しかし、実際に世の中の Web ページを見ていると、色や配置などの「表現」がざまなまに工夫されています。自分のページでもこれらの表現を行なうにはどうすればよいでしょうか? 実は過去には HTML にも「色をつける」「フォントを変える」「中央そろえ」など表現を指定する要素がありました。しかし、HTML 4.0 からはこの種の機能はすべて「非推奨」になり、代りに「スタイルシート」と呼ばれる方法で表現を指定するようになりました。

なぜでしょうか? たとえば HTML では「大見出しを全部集めて来て一覧を作る」などの作業は (`grep` などのツールを使って) 簡単に行なえますが、そのとき見出しの中に「ここは青い色」など別のタグが混ざっているとうまく取り出せなかったり、または取り出したものにタグが混ざるなど、面倒なことが起きます。

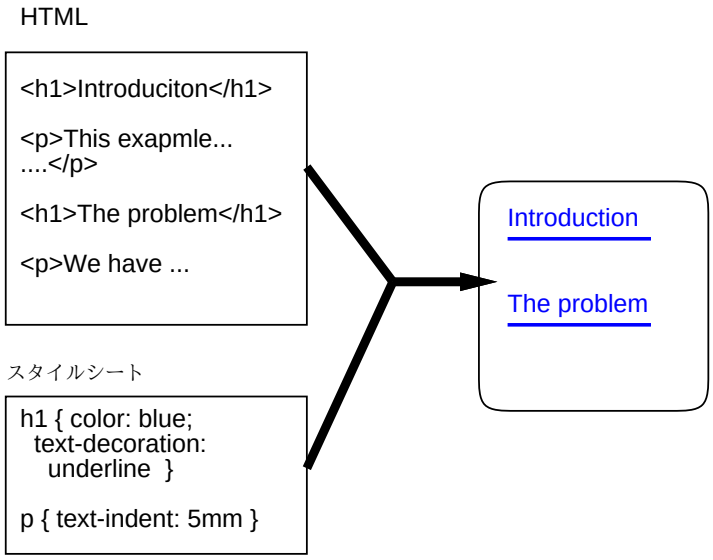

図 9: スタイルシートの概念

それに、大見出しを青い色にするとしたら、全部の大見出しをそのように統一したいわけですが、すべての大見出しの所に余分に「ここからここまで青」というタグをつけて行くのも無駄な話です。コンピュータで処理するのだから、「すべての大見出しは青」と「ひとこと」言えば済むようであるべきではないでしょうか? (図 9)。スタイルシートとはちょうどそのように、つまり文書の構造のそれぞれについて「このような部分はこのような表現」という形で表現を指定する機能なわけです。

HTML と組み合わせるスタイルシート指定言語としては **CSS**(Cascading Style Sheet) が使われます。HTML に CSS の指定を追加する方法としては、次の 3 通りがあります。

(1) CSS 指定を次のように **style 要素**の内側に書く。style 要素はヘッダ部分に入れる必要がある。

```
<style type="text/css">
```

```
CSS 指定…
…
</style>
```

(2) CSS 指定を別ファイルに入れ、HTML のヘッダ部分に次のような **link 要素**を入れる (ここでは CSS 指定が `mystyle.css` というファイルに入っているものとしました)。この方法はやや繁雑だけれど、1 つの CSS ファイルを複数の HTML ページに適用させられる。

```
<link rel="stylesheet" href="mystyle.css" type="text/css">
```

(3) HTML の各要素に **style 属性**を指定し、その値として CSS 指定の本体部分を書く。特定の要素だけに表現を指定する場合に使う。

```
<p style="color: blue">この段落は青い。</p>
```

以下では (1) の方法を使うようにします。CSS 指定を入れたページの例を示します (図 10)。

```
<!DOCTYPE html>
<html>
<head>
<title>sample</title>
<style type="text/css">
body { background: rgb(220,200,255) }
p  { background: rgb(200,255,240) }
p  { padding: 3mm 5mm }
</style>
</head>
<body>
<h2>スタイルシートとは</h2>
<p>HTML は文書の「どこからどこまでが何を意味するか」を指定します
が、その見え方 (色やあけ方など) は指定しません。</p>

<p>現在では、HTML を補完する形で「スタイルシート」と呼ばれる機能
を組み合わせることで見え方を指定します。</p>
</body>
</html>
```

2.4 CSS の指定方法

順序が逆になりましたが、CSS の指定方法について説明しましょう。まず、CSS の指定は「規則」の集まりで、1 つの規則は次の形をしています。

セレクタ { プロパティ: 値; プロパティ: 値; … }

セレクタは、とりあえず HTML のタグと思ってください。つまり「この要素はこう表現する」という指定です。**プロパティ**は、色や字下げなどです (すぐ後で説明します)。そして、それに対する値を指定します。値の指定方法は次の通り。

- 文字サイズの指定方法: `12pt` (ポイント数)、`xx-small`、`x-small`、`small`、`medium`、`large`、`x-large`、`xx-large`、百分率 (下記。本来のサイズの何倍/何%という形で指定)。
- 長さの指定方法: `1px`(画面上の点)、`1cm`(センチ)、`1em`(文字「m」の幅 1 個ぶん) などがある。
- 百分率: `%`をつけた数値。ページ幅に対する割合などの指定に使用。
- 色の指定方法: `black`、`blue`、`gray`、`green`、`maroon`、`navy`, `olive`、`purple`、`red`、`silver`、`white`、`yellow`、`rgb`(赤, 緑, 青) ただし赤/緑/青は 3 原色の強さを 0〜255 の数値で表す。

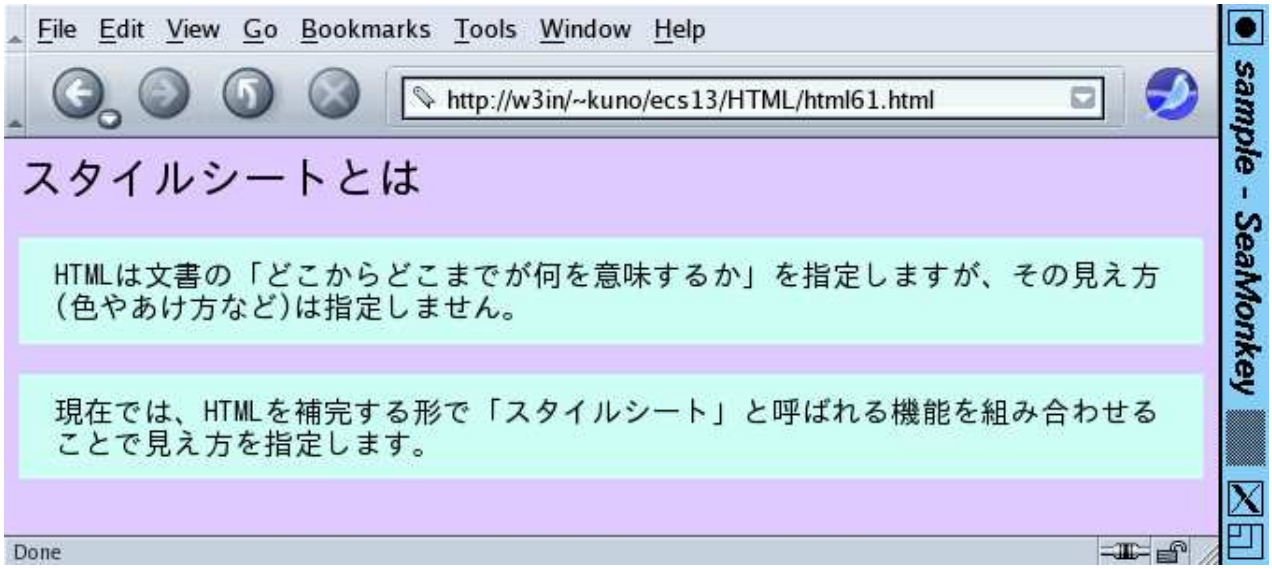

図 10: スタイルシートを使ったページ

- ファイルや URL: `url(`ファイル名`)`、`url(`URL`)`。

CSS プロパティの代表的なものとしては次のものがあります。

- `color:` 色 — 文字色を指定。
- `background:` 色 — 背景色を指定。
- `margin:` 長さ — 要素の周囲のマージン (余白) 幅を指定。4 つの長さを指定すると「上、右、下、左」の長さを指定したことになる。2 つだと「上下、左右」、1 つだと 4 周全部がその長さに。
- `padding:` 長さ — 要素の周囲のパディング (詰めもの) 幅を指定。指定方法は margin と同様。
- `border:` 形状 色 長さ — 要素の枠を指定。形状として、`solid`(均一)、`dashed`(点線)、`double`(2 重線)、`ridge`(土手)、`groove`(溝)、`inset`(くぼみ)、`outset`(出っぱり) 等が指定できる。色は枠の色、長さは枠の幅を指定。
- `text-indent:` 長さ — 段落先頭の字下げ幅を指定。
- `text-align:` 種別 — `left`、`right`、`center` で左そろえ、右そろえ、中央そろえを指定。
- `text-decoration:` 文字飾り。`underline`(下線)、`blink`(点滅) 等を指定。
- `font-size:` 文字の大きさを指定。

演習 先の例の HTML をまずそのまま表示し、次に段落や本体の背景色を変更してみなさい。その後、以下の課題から 1 つ以上 (できれば全部) 選んでやってみなさい。

a. 先頭の見出しの文字色を変更し、下線つきにしなさい。さらに、文字サイズを本来の 3 倍にしなさい。さらに太い枠線で囲みなさい。できれば、枠線がページ幅一杯でなくページの中央部 50%の範囲になるとなおよいです。[2]

b. 段落先頭の字下げや、段落の左右マージン、上下マージンを調整して、段落が連続しているときも区切りが見やすいようにするにはどのように調整すればいいかを検討しなさい。できれば、h4〜h6 のような小見出しに続いて段落がある場合についても (小見出しのスタイルも含めて) 検討できるとなおよいです。

[2]ヒント: 左右のマージンをページ幅の 25%ずつにすればいいはずですね。

c. em 要素を使って段落の一部を囲んで強調の様子を見てみなさい。さらにそれがより自然なようすに見えるように調整を試みなさい。できれば、a 要素についてもスタイルを変更して同様に検討してみるとなおよいでしょう。[3]

2.5 HTML のフォームと GUI 部品

前節で述べたように、HTML でユーザインタフェースを構築する場合には form 要素と GUI 部品を利用します。まず form 要素から説明しましょう。

- `<form name="名前" method="手法" action="`*URL*`"> ... </form>` — フォームはサーバにデータを送る「かたまり」を表し、その中に含まれる GUI 部品の値がまとめて送られることになる。`name` はページ内スクリプトからフォームを参照する時に使う。[4]

form 要素の内側には、さまざまな入力部品 (HTML 用語ではコントロールと言います) を入れます。入力部品は通常のテキストや HTML 要素と混ぜて入れられるので、通常の HTML の機能を使って説明文やラベルをつけたり、見やすく配置することができます。

入力部品は以下で説明するようにさまざまな種別がありますが、すべてに共通するのは `name` 属性 (名前) と `value` 属性 (値) です。これらは、サーバ側にフォームが送られる時に対になって送られます。たとえば「`name="age" value="30"`」という属性だったら、「age=30」という対が送信されるわけです。なお、値は部品によってはユーザが入力した文字列になります。

以下に主要な入力部品とその属性指定を説明します (`name` と `value` は上記の通りなので特に注記することがない場合は説明していません。また☆はすぐ後の例題に出て来るものです)。

☆ `<button [name="名前"] [value="値"]>...</button>` — 送信ボタン。要素の内容がボタン内に表示される。ボタンが押されるとフォーム内容が送信される。

☆ `<input type="text" name="名前" [size="長さ"] [value="値"]>` — テキスト入力欄。送信時にその内容が送られる。`size` で欄幅 (文字数)、`value` で初期値を指定可能。

- `<input type="password" name="名前" [size="長さ"] [value="値"]>` — 上と同じだが、表示は「***…」となる。ただしデータが暗号化されるわけではないので、本当に盗まれるとまずい情報は暗号化通信を使ったページで扱うこと。
- `<textarea [rows="行数"] [cols="文字数"]> ... </textarea>` — 複数行入力欄。機能は textarea と同様要素の中身が初期値になる。`rows`、`cols` で大きさを指定。
- `<input type="checkbox" name="名前">` — チェックボックス。画面上でチェックを ON/OFF でき、チェックされているものだけが「`"on"`」という文字列を送信。
- `<input type="radio" name="名前" value="値" [checked]>` — ラジオボタン。同じ名前のものが複数あってよく、その中で 1 つだけが ON になる。最初に ON であって欲しいものがあればそれに `checked` を指定する。ON になっているものの値が送信される。
- `<input type="hidden" name="名前" value="値">` — 特別な部品で、画面に表示されない (従ってユーザが値を変更することもない)。常に指定された名前と値を送信する。

☆ `<select name="名前">...</select>` — 選択メニュー。内側には次に示す option 要素を複数入れられ、これらを項目とするメニューができる。

☆ `<option [value="値"] [selected]> ... </option>` — 選択メニューの項目で、選択されている時は select の値としてこの項目の `value` 値 (`value` を省略すると要素の内容) が送られる。最初に選択された状態であってほしい項目に `selected` を指定する。

---

[3] ただし、実際に Web サイトで使う時には、a 要素のスタイルを変更してしまうとリンクだということが分かりづらくなるので、やらない方がいいとされています。このことも含めて確認してください。

[4] `method` としてはデータが URL にくっついて見える「`"get"`」と別途送られる「`"post"`」のいずれかを指定しますが、省略すると「`"get"`」になります。`action` では、送信先 URL(データを処理するサーバ上のプログラムのありか) を指定します。省略すると、本のページと同じ URL が使われます。

図 11: フォーム部品の入ったページ

では実際に、これらのいくつかを使った HTML の例を見てみましょう (図 11)。

```
<!DOCTYPE html>
<html><head>
<title>sample</title>
<style type="text/css">
form { background: rgb(200,255,235); padding: 1em }
</style>
</head><body>
<h1>フォーム部品</h1>
<div><form name="data">
名字: <input type="text" name="last"><br><br>
性別: <select name="sex">
      <option value="m">♂</option>
      <option value="f" selected>♀</option></select><br><br>
<button name="btn" value="send">送信します</button>
</div></form></body></html>
```

これにより、入力欄、選択メニュー、ボタンを持つ Web ページができます。form で method を指定してないため get が用いられますが、この場合フォームを送信すると同じ HTML が表示されます。ただしそのとき、URL の末尾に名前と値の対が付加されることがブラウザの URL 表示窓で確認できます。

演習 「摂氏」「華氏」の相互変換 (温度換算) を行うページを作りたいものとします。HTML の GUI 部品を使って作ってみなさい (製作の前に紙でデザインをスケッチしてから始めること)。

3 まとめ

本日は前半で情報システムとは何かから始め、情報システムの定式化の手法である構造化分析やその図法である DFD について扱った。定式化の定石とかは知っておくと役に立つと思う。後半では HTML+CSS と HTML 上の GUI 部品機能について扱った。これらはこの後、Web アプリケーションを扱う上で必要になる。